VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce chodců pomocí příznakového rozpoznání

Pedestrian Detection using Feature-Based Detectors

2015 Jakub Kolder

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jakub Kolder**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Detekce chodců pomocí příznakového rozpoznání
Pedestrian Detection using Feature-Based Detectors

Zásady pro vypracování:

S rozvojem kamerových systémů se v posledních letech zvýšilo i jejich využití v oblasti detekce lidí (chodců). Detekce za pomocí kamerových systémů je využívána například na letištích a na dalších místech, kde je důležitá kontrola nebezpečného pohybu chodců. Neméně důležité je i využití takového detektoru v automobilech, které jsou v posledních letech hojně vybavovány kamerovými systémy pro detekci chodců v okolí automobilů. Cílem této práce bude otestování renomovaných metod pro detekci objektů právě v aplikaci detekce lidí v obrazech.

1. Seznamte se se základními pojmy v oblasti detekce chodců v obrazech.
2. V popisu se zaměřte zejména na příznakové rozpoznávání pomocí renomovaných příznaků (např. HOG, Haar, LBP).
3. Seznamte se s knihovnou OpenCV.
4. S pomocí knihovny OpenCV tyto detektory implementujte.
5. Experimentálně ověřte funkčnost, přesnost a rychlost těchto detektorů na renomovaných datasetech.
6. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

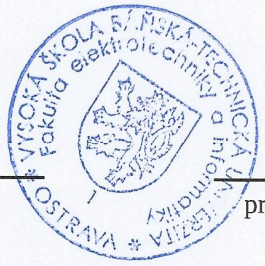
[1] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1, pp. I-511-I-518 vol.1 (2001)
[2] Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: ICB. pp. 828-837 (2007)
[3] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. vol. 1, pp. 886-893 vol. 1 (june 2005)

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2015



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2015 .....

Tímto bych rád poděkoval svému vedoucímu, Ing. Radovanu Fuskovi, za poskytnuté odborné rady, za neocenitelné zkušenosti a za všechen čas, jenž mi takto věnoval.

Abstrakt

Tato práce je zaměřená na vytvoření detektorů chodců v obraze s využitím příznakového rozpoznávání renomovaných příznaků. Nejprve jsou uvedený základní principy příznakových metod, kterými jsou histogramy orientovaných gradientů, Haarovy příznaky a metoda lokálních binárních vzorů. V další části jsou představeny klasifikátory Support Vector Machines a Adaptive Boost. Výstupem této práce je implementace těchto metod s pomocí knihovny OpenCV a experimentálně ověřit a porovnat jejich funkčnost.

Klíčová slova: extrakce příznaků, detekce objektů, HOG, Haar, LBP, SVM, AdaBoost

Abstract

This thesis is focused on creating pedestrians detectors in image using pattern recognition renowned features. At first are introduced the basic principles of feature-based methods, which are histograms of oriented gradients, Haar-like features and method of Local Binary Patterns. The next part are introduced classifiers Support Vector Machines and Adaptive Boost. The result of this thesis is the implementation these methods using the OpenCV library and experimentally verify and compare their functionality.

Keywords: feature extraction, object recognition, HOG, Haar, LBP, SVM, AdaBoost

Seznam použitých zkratek a symbolů

HOG	–	Histogramy orientovaných gradientů
Haar	–	Haarovy příznaky
LBP	–	Local Binary Pattern
SVM	–	Support Vector Machines
AdaBoost	–	Adaptive Boost

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

S rozvojem kamerových systémů se v posledních letech zvýšilo i jejich využití v oblasti detekce lidí (chodců). Detekce za pomocí kamerových systémů je využívána například na letištích a na dalších místech, kde je důležitá kontrola nebezpečného pohybu chodců. Neméně důležité je i využití takového detektoru v automobilech, které jsou v posledních letech hojně vybavovány kamerovými systémy pro detekci chodců v okolí automobilů. Cílem této práce je vytvoření a otestování renomovaných metod pro detekci objektů právě v aplikaci detekci lidí v obrazech. První polovina této práce je zaměřená na popisem příznakových metod a jejich klasifikací. Metoda založena na histogramech orientovaných gradientů je první příznakovou metodou, která je poté využita v klasifikátoru SVM. Dalšími metody jsou lokální binární vzory a Haarovy příznaky, kde výběr jednotlivých příznaků je realizován algoritmem AdaBoost, u kterých zvýšení detekční rychlosti a snížení počtu falešných detekci je zaveden koncept kaskády klasifikátorů. V druhé polovině práce, jsou popsány procesy trénování jednotlivých příznakových metod a jejich aplikováni na sadě testovacích snímků. Výsledky experimentů jednotlivých metod jsou zdokumentovány a porovnány.

2 Příznakové rozpoznávání

Obrazy obsahují objekty, které lze zařadit do jednotlivých tříd (chodec, obličej, automobil, jablko atd.). Účelem příznakového rozpoznávání je pro popis těchto objektů extrahovat příznaky, které vykazují určité hodnoty. Tyto hodnoty se musí natolik lišit, aby klasifikátor dokázal správně rozlišit do jaké třídy je objekt zařazen. V této práci pro extrakci příznaků a jejích klasifikaci je použito několik metod.

2.1 Histogramy orientovaných gradientů

HOG příznaky zavedli Navneed Dalal a Bill Triggs[1], kteří vyvinuli a otestovali několik variant HOG deskriptorů. Ve své práci vyzkoušeli různé druhy gradientních operátorů, různé normalizační metody a ukázali jak správně nastavit další parametry této detekční metody v aplikaci detekování chodců.

Základní myšlenka metody HOG je, že objekt v obraze může být pomocí vzhledu a tvaru charakterizován intenzitou gradientů, nebo směrem hran. Obraz se rozdělí na malé prostorové oblasti (buňky) a pro každou buňku se sestaví histogram orientovaných gradientů, který je vypočítán ze všech pixelů z buňky. Je vhodné obraz před započetím výpočtů normalizovat, například kontrastní normalizací, nebo normalizací osvětlení. Toho lze docílit shromažďováním informací do histogramu nejen z jedné konkrétní buňky, ale z větší oblasti z okolních buněk. Těchto několik buněk dá dohromady tzv. blok.

Obrázek 1: Algoritmus metody HOG.

2.1.1 Histogram

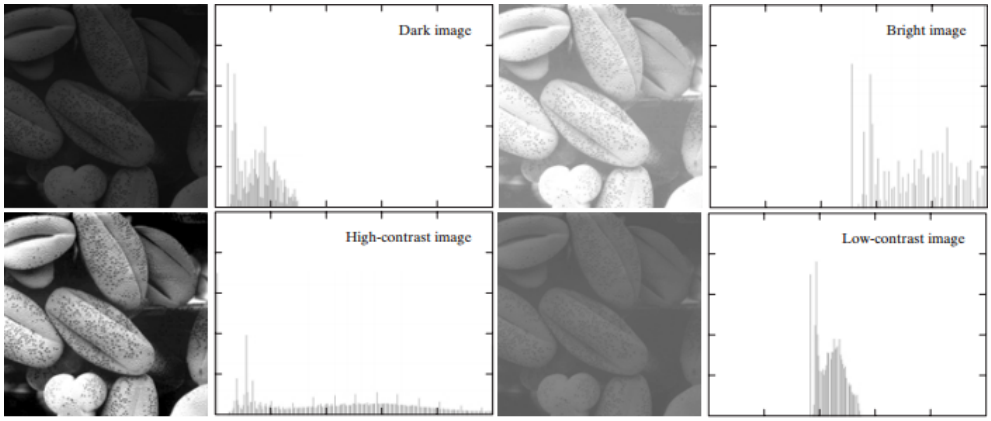
Histogram je sloupcové zobrazení intervalových četností, kdy šířka sloupce je šířkou intervalu a výška sloupce představuje četnost výskytu dané třídy. Histogram digitálního obrazu s úrovni šedi v rozsahu $[0, L-1]$ je diskrétní funkce[5]:

$$h(r_k) = n_k, \tag{2.1}$$

kde r_k je k-tá úroveň šedi a n_k je počet pixelů v obraze s intenzitou šedi r_k.[5] Běžně se používá normalizace histogramu pomocí vydělení jednotlivé složky celkovým počtem obrazových bodů v obraze n. Normalizovaný histogram je tedy dán[5]:

$$p(r_k) = \frac{n_k}{n}, \tag{2.2}$$

pro $k = [0, 1, ..., L-1]$. Volně řečeno, $p(n_k)$ je tedy odhad pravděpodobnosti výskytu úrovně šedi r_k v obraze. Součet všech prvků normalizovaného histogramu je 1. Histogram obrazu s úrovněmi šedé intenzity má čtyři základní charakteristiky: Tmavé, světlé, s nízkým kontrastem a s vysokým kontrastem. Ukázka na obrázku [2].

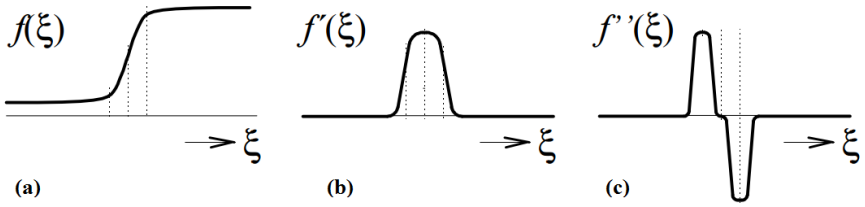


Obrázek 2: Čtyři základní typy histogramu.[5]

2.1.2 Detekce hrany

Každý objekt je v obraze reprezentován souvislou oblastí. Každá oblast je obklopena hranicí. Hranice se skládá z hran (případně též z jediné zakřivené hrany). Hrana se skládá z jednotlivých hranových bodů. Za bod hrany se nejčastěji považuje místo, kde průběh jasu vykazuje náhlou změnu, případně inflexní bod. Po nalezení jsou jednotlivé nalezené body hran spojovány různými technikami do hran a celých hranic.

Významným problémem v analýze obrazů je nalezení hran a celých hranic. Při řešení se často ale postupuje tak, že se nejprve naleznou jednotlivé body hran. Uvažujme pro jednoduchost obrazy ve stupních šedi. (I v praxi se před analýzou, a tedy i před hledáním hran, barevné obrazy velmi často převádí na obrazy ve stupních šedi.) Pro existenci bodu hrany se nejčastější využívá hledáni maxima prvních derivací nebo hledání průchodu druhých derivací nulou. Na obrázku [3]: (a) je znázorněn typický průběh jasu napříč hranou, (b) první derivace a (c) průchod druhé derivace nulou, kterou si nebudeme popisovat.[6]



Obrázek 3: Průběh jasu a jeho první a druhé derivace ve směru napříč hranou.[6]

2.1.3 Výpočet gradientů

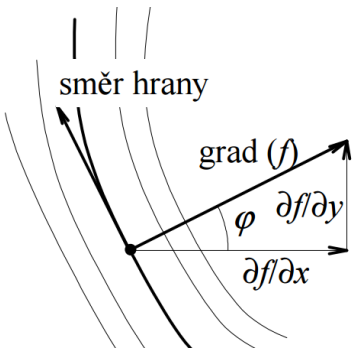
Prvním krokem algoritmu je výpočet gradientů. Gradient se v oblasti zpracování obrazu používá pro detekci hran objektů a směru v místě (x, y). Gradient f, v bodě (x, y) označený $\bigtriangledown f$, je definován jako vektor[11]:

$$\bigtriangledown f = \text{grad}(f) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.3}$$

Směr, kde je změna jasu největší je gradient obrazové funkce. Směr hrany je pak kolmý ke směru gradientu (Ukázka na obrázku [4]). Za velikost hrany lze vzít velikost gradientu. Obrázek [3] ukazuje typický průběh jasu ve směru napříč hranou a jeho první, i druhou derivací tohoto průběhu.[5][6] Gradientní metody využívají skutečnosti, že v místě hrany má absolutní hodnota první derivace průběhu jasu vysokou hodnotu. Hodnota derivace popisuje intenzitu kontury v daném bodě, tedy velikost hrany. Hranové operátory stanovují velikost hrany. Nejjednoduššími hranovými operátory jsou zřejmě derivace $\partial f/\partial x$ a $\partial f/\partial y$, které popisují změnu úrovně jasu ve směru os x a y. Tyto operátory je možné použit v případě, že hrany jsou rovnoběžné s osami, ale při hledání hran obecného směru je potřeba vyšetřovat průběh jasu v kolmém směru na směr potenciální hrany. Použitím derivace a vektoru, popisující směr kolmý k potenciální hrany, lze ověřit zda v daném bodě existuje hrana. Takovýto vektor je $\mathbf{n} = (\cos\theta, \sin\theta)$ a pro derivaci ve směru kolmo k hraně je vzorec[6]:

$$\frac{\partial f}{\partial \xi} = \text{grad}(f) \cdot \mathbf{n} = \frac{\partial f}{\partial x}\cos\theta + \frac{\partial f}{\partial y}\sin\theta, \tag{2.4}$$

kde $|\partial f/\partial \xi|$ je velikost hrany v daném bodě a ξ je souřadnice měřená v tomto směru.



Obrázek 4: Stanovení velikosti a směru hrany na základě gradientu.[6]

Protože už víme, že směr gradientu je kolmý ke směru hrany a za velikost hrany, označený $e(x, y)$, lze vzít velikost gradientu, tak na základě rovnice (2.4) lze určit směr potenciální hrany. Pro stručnost je zavedeno označení $f_x(x, y) = \partial f(x, y)/\partial x, f_y(x, y) =$

$\partial f(x,y)/\partial y$. S tímto označením jsou dány vzorce[6]:

$$e(x,y) = \sqrt{f_x^2(x,y) + f_y^2(x,y)}, \tag{2.5}$$

$$\varphi(x,y) = \arctan\left[\frac{f_y(x,y)}{f_x(x,y)}\right], \psi(x,y) = \varphi(x,y) + \frac{\pi}{2}, \tag{2.6}$$

kde $\varphi(x,y)$ je směr gradientu a $\psi(x,y)$ je směr hrany v bodě (x,y).

Nejjednodušším rozhodnutím, zda vyšetřovaný bod leží na hranici nějakého objektu, je případ, kdy hodnota $e(x,y)$ je větší než předem zvolena prahová hodnota. Tento postup má své nedostatky, kde hranice objektu vyjde širší než jeden bod a neřeší problémy nespojité funkce. Jelikož tento postup předpokládá, že obrazová funkce je spojitá. Nejpraktičtějším využitím je pracovat s diskrétní funkcí. Nahrazení derivace diferencemi, lze vztahy (2.5) a (2.6) použít pro diskrétní případ. Diference lze napsat[6]:

$$f_x(x,y) = f(x+1,y) - f(x,y), \tag{2.7}$$

$$f_y(x,y) = f(x,y+1) - f(x,y). \tag{2.8}$$

Praktická realizace výpočtu je hodnota $e(x,y)$ stanovována a porovnávána s hodnotou prahu ve všech bodech obrazu. Při stanovení délky gradientu lze použít i jiných metrik, než je metrika uvedená ve vztahu (2.5). Lze použít např. následujících vztahů[6]:

$$e(x,y) = |f_x(x,y)| + |f_y(x,y)|, e(x,y) = max\{|f_x(x,y)|, |f_y(x,y)|\}. \tag{2.9}$$

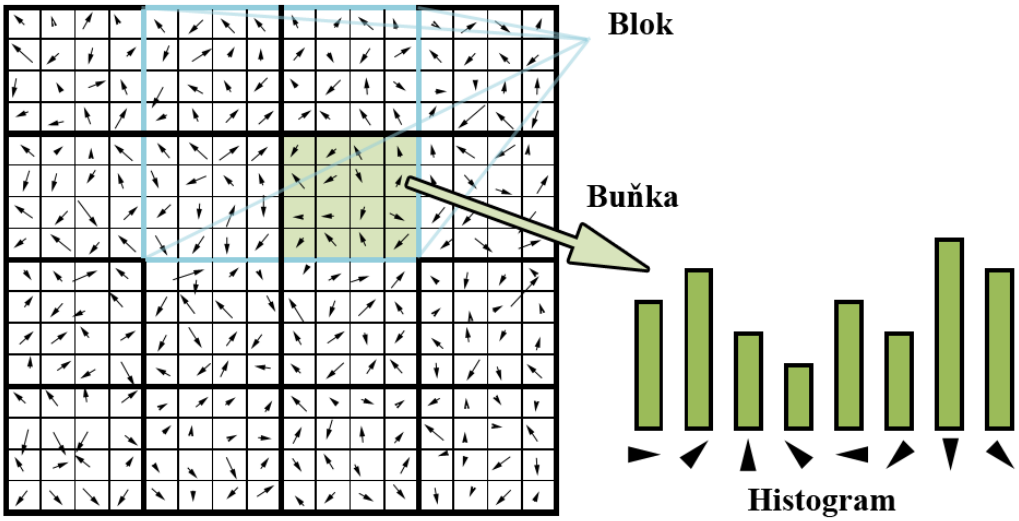
V minulosti bylo realizováno mnoho praktických metod pro detekci hran, jejichž společným teoretickým základem je výpočet velikosti gradientu. Příklady hranových operátorů: Roberts, Prewitt, Sobel, Robinson, Kirsch, Laplacián. Zmíněné operátory se odlišují především v citlivosti na šum a vhodnosti pro detekci odlišných typů hran. Autoři článku [1] při testování těchto hranových operátorů nakonec zjistili, že nejlepší metodou je použít 1-D masku, střed bodu diskrétní derivace v obou směrech, horizontální i vertikální. Tato metoda vyžaduje filtrování ve stupních šedi s těmito jádrovými filtry[11]:

$$D_X = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ and } D_Y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{2.10}$$

Jestliže, je dán obrázek I dostaneme X-horizontální a Y-vertikální deriváty použitím konvoluce: $I_X = I * D_X$ a $I_Y = I * D_Y$, tak výpočet gradientu je $|G| = \sqrt{I_X^2 + I_Y^2}$ a orientace gradientu je dána: $\theta = \arctan \frac{I_Y}{I_X}$

2.1.4 Rozdělení obrazu na buňky

Dalším krokem metody HOG je rozdělení obrazu na buňky. Autoři v článku [1] dosáhli nejlepších výsledků s použitím buněk o velikosti 8×8 pixelů. Nyní máme obraz rozdělený na buňky a je potřeba pro každou buňku zvlášť určit histogram orientovaných gradientů. Kanály histogramu každé buňky jsou dány směrem gradientů a velikost kanálu je dána velikosti gradientů. V článku [1] autoři rozdělili buňku na 9 kanálů, ale buňka může být rozdělená na libovolný počet kanálů. Rozsah kanálů je $0° - 180°$, nebo $0° - 360°$. Po prozkoumání orientace každého gradientu v buňce se zařadí do určitého kanálu. Výsledná velikost kanálu je dána součtem gradientů zařazených do stejného kanálu. Ukázka kanálů je na obrázku [5].



Obrázek 5: Histogram orientovaných gradientů pro každou buňku.

2.1.5 Bloky deskriptoru a jejich normalizace

HOG Deskriptor je vektor normalizovaných histogramů získaný z bloku. Z každého bloku je získán jeden deskriptor. Bloky se obvykle překrývají, což znamená, že každá buňka se může podílet na více deskriptorech. Deskriptory jsou předány nějakému klasifikátoru.

Nerovnoměrným osvětlením a různým kontrastem se liší velikost gradientů v celém obraze. Normalizace je založená na odstranění těchto vlivů. Autoři [1] zkoušeli různá normalizační schémata. Většina z nich je založená na normalizaci každého bloku zvlášť. Překrývání jednotlivých bloků je výhodnější, protože jednotlivé buňky budou zpracovávány vícekrát. Bloky existují obdélníkové (R-HOG) a kruhové (C-HOG). R-HOG bloky jsou obecně čtvercové. Taktéž normalizace může být provedená čtvercová, nebo kruhová. Dalal a Triggs testováním zjistili, že nejvýhodnější velikost buněk čtvercových bloků, které převážně používali, je 2×2, nebo 3×3. Existuje mnoho metod pro normalizaci,

některé si uvedeme. Nechť v je nenormalizovaný vektor obsahující všechny histogramy v daném bloku. $||v_k||$ bude jeho k-normou pro $k = 1, 2$ a e bude malá konstanta, která neovlivní výsledek. Potom normalizační výsledek může být jeden z následujících[1]:

$$\text{L2-norm: } f = \frac{v}{\sqrt{||v||_2^2 + e^2}} \tag{2.11}$$

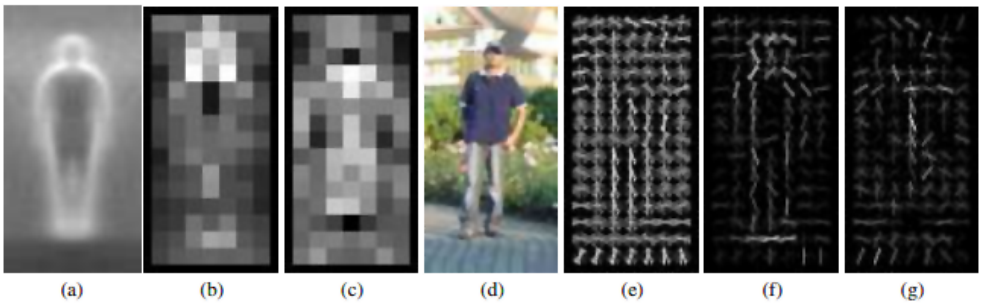
$$\text{L1-norm: } f = \frac{v}{||v||_1 + e} \tag{2.12}$$

$$\text{L1-sqrt: } f = \sqrt{\frac{v}{||v||_1 + e}} \tag{2.13}$$

2.1.6 Detekční okno

V článku [1] autoři zjistili, že detekční okno o velikosti 64×128 zahrnuje okraj o velikosti asi 16 pixelů na každé straně od postavy. Tato hranice poskytuje značné množství kontextu, která pomáhá detekci. Snížením velikosti okraje na každé straně z 16 na 8 pixelů (detekční okno o velikosti 48×112) zhoršilo výsledky o 6%. Zachováním velikosti okna 64×128, ale zvýšením velikosti postavy (opět snížení hranice) způsobí podobnou ztrátu výkonu.

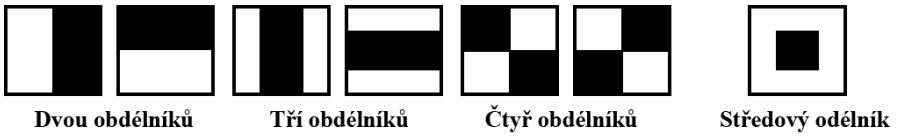
Na obrázku [6] je ukázka HOG detektorů z publikace [1], který znázorňuje především obrys postavy. Nejaktivnější bloky se soustředí na pozadí obrazu těsně mimo obrys. (a) Průměr obrazového gradientu přes trénovací příklady. (b) Každý "pixel"znázorňuje maximální pozitivní váhu SVM v bloku se středem v pixelu. (c) Stejně tak pro negativní váhu SVM. (d) Testovací snímek. (e) Vypočítán R-HOG deskriptor. (f,g) Vážený R-HOG deskriptor, resp. pozitivní a negativní váhy SVM.



Obrázek 6: HOG detektory.

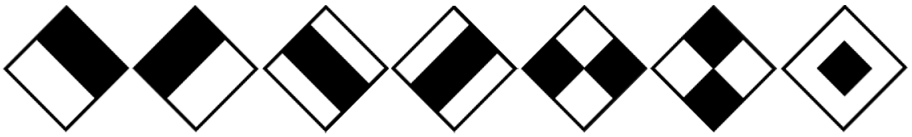
2.2 Haarovy příznaky

Detektor s Haarovými příznaky prvně představili autoři článku [2] Paul Viola a Michael Jones. Tento příznakový systém pracuje mnohem rychleji a přesněji, a je založený na rozdílu jasu mezi obdélníkovými oblastmi. Konkrétně používá několik druhu příznaků: Příznaky *dvou obdélníků*, kde hodnota příznaku je rozdíl mezi součtem pixelů dvou obdélníkových oblasti. Tyto oblasti mají stejnou velikost a tvar a jsou horizontálně, nebo vertikálně sousedící. Ukázka na obrázku [7]. Druhým příznakem je příznak *tří obdélníků*, která počítá součet dvou vnějších obdélníků a odečte od součtu středového obdélníku. Další příznak počítá rozdíl mezi diagonálními páry obdélníků, tento příznak je příznak *čtyř obdélníků*. Autoři článku použili výše zmíněné příznaky, ale existuje ještě příznak středového obdélníku. Ukázka na obrázku [7].



Obrázek 7: Haarovy příznaky přímé

Tyto příznaky jsou tzv. základní set (Basic Haar Set), ten byl následně rozšířen o příznaky natočené o 45 stupňů. Viz. obrázek [8].



Obrázek 8: Haarovy příznaky natočené

Proces generování příznaků nejprve nastaví nejmenší možný rozměr příznaku a následně je příznak posouván oknem horizontálně, nebo vertikálně o jeden pixel. Poté je příznak přidán do seznamu všech příznaků. Při posunu do konce okna je příznak zvětšen a proces se opakuje, dokud příznak není větší než velikost okna. Základní řešení detektoru v článku [2], je velikost detekčního okna 24×24 pixelů. Autoři zde detekují obličej. V jejich případě je celkový počet vygenerovaných příznaků přes 180 000. V této práci detekujeme chodce a optimální velikost detekčního okna je 24×48 pixelů. Tedy v naší práci bude celkový počet příznaků více než dvakrát tak větší. Pro rychlou funkci a natrénováni detektoru slouží integrální obraz, který je popsán v následující části a metody kaskáda klasifikátorů a AdaBoost, které jsou podrobně popsány v kapitole [3].

2.2.1 Integrální obraz

Obdélníkové příznaky lze počítat velmi rychle pomocí zprostředkující reprezentace pro obraz, který se nazývá integrální obraz. Integrální obraz na souřadnicích x a y, obsahuje součet pixelů nad a vlevo od x a y včetně[2]:

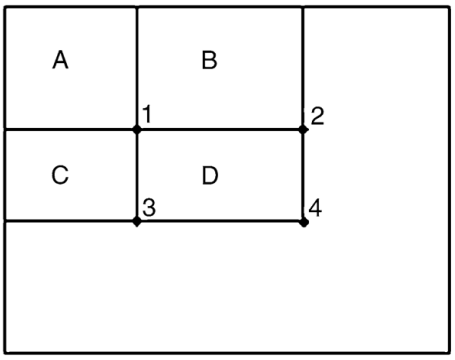
$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'), \tag{2.14}$$

kde $ii(x, y)$ je integrální obraz a $i(x, y)$ je originální obraz. Pomocí následující dvojce vzorců[2]:

$$s(x, y) = s(x, y - 1) + i(x, y), \tag{2.15}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y), \tag{2.16}$$

kde $s(x, y)$ je kumulativní součet hodnot řádku $s(x, -1) = 0$ a $ii(-1, y) = 0$. Integrální obraz může být vypočítán v jednom průchodu přes původní obrázek.

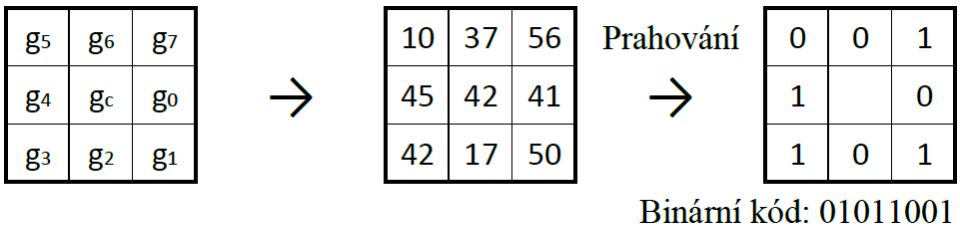


Obrázek 9: Výpočet hodnoty integrálního obrazu.[2]

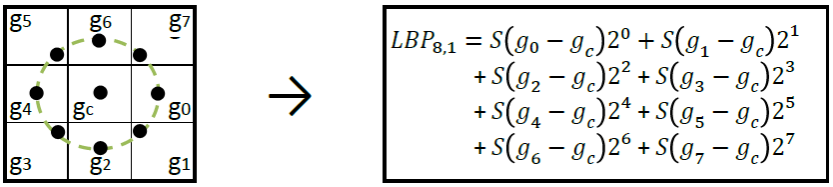
Součet pixelů v obdélníku D, lze počítat se čtyřmi odkazy pole. Hodnota integrálního obrazu v místě 1 je součet pixelů v obdélníku A. Hodnota v místě 2 je A + B, na místě 3 je A + C a na místě 4 je A + B + C + D. Součet D může být počítán jako 4 + 1 - (2 + 3). Pomocí integrálního obrazu suma nějakého obdélníku může být vypočtena čtyřmi odkazy pole (viz obrázek [9]). Je zřejmé, že rozdíl mezi sumy dvou obdélníků lze vypočítat v osmi odkazech. Vzhledem k tomu, že příznaky dvou obdélníků definované výše zahrnují sumy přilehlých obdélníků, lze rozdíl vypočítat v šesti odkazech pole, osm v případě funkcí tří obdélníků a devět pro funkcí čtyř obdélníků.[2]

2.3 Local Binary Pattern

Local Binary Pattern, neboli lokální binární vzory, zavedl Timo Ojala.[3] Standardní verze LBP jednoho pixelu je tvořen prahováním každého pixelu s hodnotou středového pixelu na okolí pole o velikosti 3×3. Nechť g_c je středový pixel šedého odstínu a $g_i(i = 0, 1, ..7)$, je šedá úroveň každého okolního pixelu. Obrázek [10] znázorňují základní operace LBP. V případě, že g_i je menší než g_c, je binární výsledek pixelu nastavena na 0 jinak je nastaven na 1. Všechny výsledky jsou kombinovány k dostání 8 bitové hodnoty. Binární hodnota desítkové soustavy je LBP příznak.



Obrázek 10: Základní LBP operátor.



Obrázek 11: LBP operátor kruhového charakteru.

Metoda bilineární interpolace se používá pro vzorkování bodu, který nepatří do středového pixelu. Nechť $\text{LBP}_{p,r}$ značí LBP příznak kruhového charakteru, kde r je poloměr a p je počet sousedních bodů na kruhu. Z obrázku [11] lze napsat vzorec[3]:

$$LBP_{p,r} = \sum_{p=0}^{p-1} s(g_i - g_c)2^i, S(x) = \begin{cases} 1 & \text{jestliže } x \leq 0 \\ 0 & \text{v opačném případě} \end{cases} \tag{2.17}$$

LBP je v odstínech šedé neměnný a rotačně neměnný. Tato vlastnost je vhodná i pro mnoho aplikací. LBP je vhodné pro detekci základních obrazových primitiv jako jsou špičky, konce čar, hrany, rohy. Typ detekované primitivy je závislý na konfiguraci LBP vzoru. Pro zajištění rotační neměnnosti je nutné příznaky normalizovat. Normalizace vychází z předpokladu, že okolní body lze (po prahování a váhování) vyjádřit jako binární vzor, který je možné rotovat (nejčastěji doprava).[11]

3 Klasifikace příznaků

3.1 Support Vector Machines

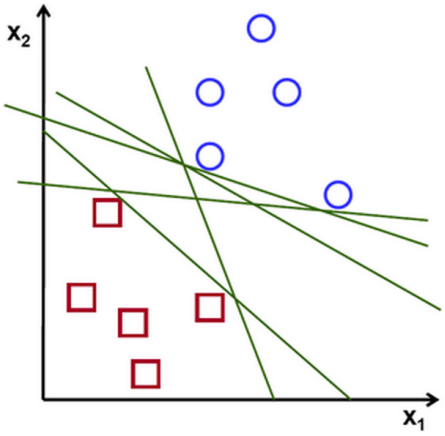
Zásadní součástí strojového učení SVM je jádrová transformace (angl. kernel transformation) prostoru příznaků dat do prostoru transformovaných příznaků typicky vyšší dimenze. Tato jádrová transformace umožňuje převést původně lineárně neseparovatelnou úlohu na úlohu lineárně separovatelnou, na kterou lze dále aplikovat optimalizační algoritmus pro nalezení rozdělující nadroviny. SVM byl původně diskriminační klasifikátor. V úloze binární klasifikace SVM hledá nadrovinu, která v prostoru příznaků optimálně rozděluje trénovací data do dvou tříd (pozitivní a negativní vzorky). Později SVM byla rozšířena o regresní a clustering problémy.[7]

3.1.1 Lineárně oddělitelná data

Optimální nadrovina (angl. optimal hyperplane) je taková, že body leží v opačných poloprostorech a minimální hodnota vzdálenosti bodů od roviny je co největší. Taky by se dalo říct, že okolo nadroviny je na obě strany co nejširší oblast bez bodů tzv. maximální odstup (angl. maximal margin). Na popis nadroviny stačí pouze nejbližší body, kterých je obvykle málo. Tyto body se nazývají podpůrné vektory (angl. support vectors) a odtud název metody. Optimální nadrovina je dána vztahem[7]:

$$f(x) = \beta^T x + \beta_0, \tag{3.1}$$

kde β je váhový vektor a β_0 je práh. Optimální nadrovina je dána ve 2D prostoru přímkou. Tato přímka je získaná, na základě optimalizačního algoritmu, z možnosti existence několika rozdělovacích přímek, které jsou ukázaný na obrázku [12]. Optimalizační algoritmus říká, že optimální nadrovina má největší minimální vzdálenost od trénovacích dat, protože rozdělovací přímky probíhající příliš blízko bodů jsou náchylné na šum a nebudou správně fungovat. Ve 3D prostoru je rozdělovačem rovina.



Obrázek 12: Rozdělení dvou tříd přímkou.[7]

Optimální nadrovina může být reprezentována v nekonečném počtu různých způsobů, změna velikosti zvyšováním váhového vektoru β a prahu β_0. Proto, mezi všemi možnými reprezentacemi nadroviny, byl vybrán základní předpis[7]:

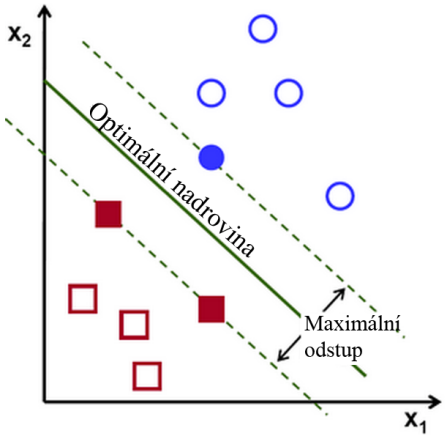
$$|\beta^T x + \beta_0| = 1, \tag{3.2}$$

kde x symbolizuje podpůrné vektory. Na obrázku [13] jsou tyto podpůrné vektory označeny plnými čtverci a plným kruhem. Tato reprezentace je známa jako kanonická nadrovina. S využitím kanonické nadroviny je možné vypočítat vzdálenost podpůrných vektorů od nadroviny vztahem[7]:

$$\text{vzdálenost} = \frac{|\beta^T x + \beta_0|}{||\beta||} = \frac{1}{||\beta||}, \tag{3.3}$$

kde dvojnásobek této vzdálenosti je maximální odstup a tvoří pásmo znázorněno v obrázku [13].

$$M = \frac{2}{||\beta||} \tag{3.4}$$



Obrázek 13: Rozdělovací nadrovina a odstup.

Konečný, problém maximalizace M je ekvivalentní problému minimalizace funkce $L(\beta)$podléhající některým omezením. Omezení modelu je požadavek na nadrovinu správně klasifikovat všechny podpůrné vektory x_i. Formálně[7]:

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \textit{ v závislosti na } y_i(\beta^T x_i + \beta_0) \geq 1 \forall i, \tag{3.5}$$

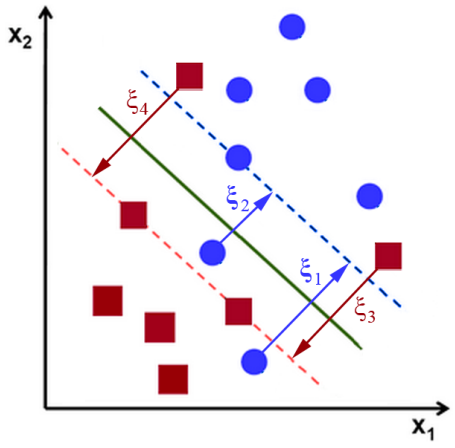
kde y_i představuje označení všech bodu trénovacích dat. To je problém Lagrangeovy optimalizace, která lze řešit pomocí Lagrangeových multiplikátorů k získaní hmotnosti vektoru β a vychýlení β_0 optimální nadroviny.[8]

3.1.2 Nelineárně oddělitelná data

Problémem optimálního rozdělovače je skutečnost, že trénovací data můžou být jen zřídka lineárně oddělitelná. Je nutné si připustit, že bude nalezena nadrovina se špatnou klasifikací. Tato špatná klasifikace je nová proměna v optimalizaci, která musí být brána v úvahu. Nový model zahrnuje obě úlohy, jak nalezení optimální nadroviny, která dává maximální odstup, tak generalizaci správných trénovacích dat s umožněním příliš mnoho klasifikačních chyb. Vycházíme z formulace problému optimalizace pro vyhledávání nadroviny, která maximalizuje odstup. Tato formulace je zmíněná ve vzorci (3.5). Model může být několika způsoby upraven tak, že bere v úvahu chybnou klasifikaci. Dobrým řešením je vzít v úvahu vzdálenosti od chybných vzorků ke správné rozhodující oblasti, tedy[9]:

$$\min||\beta||^2 + C \tag{3.6}$$

Parametr C je regulační parametr, který reguluje velikost maximálního odstupu a množství klasifikačních chyb. Jestliže, pro parametr C je zvolena vysoká hodnota, tak bude výsledkem menší počet klasifikačních chyb, ale získaný odstup bude menší. V případě zvolení nízké hodnoty bude získaný odstup větší, ale také bude větší množství klasifikačních chyb. Jak je ukázáno na obrázku [14], který ukazuje nelineárně oddělitelné trénovací data ze dvou tříd, oddělovací nadroviny a vzdálenosti správné oblasti, které jsou chybně, tak u každého vzorku trénovacích dat je definován nový parametr ξ_i. Každý z těchto parametrů obsahuje vzdálenost odpovídající natrénovaného vzorku se správnou rozhodující oblasti.[9]

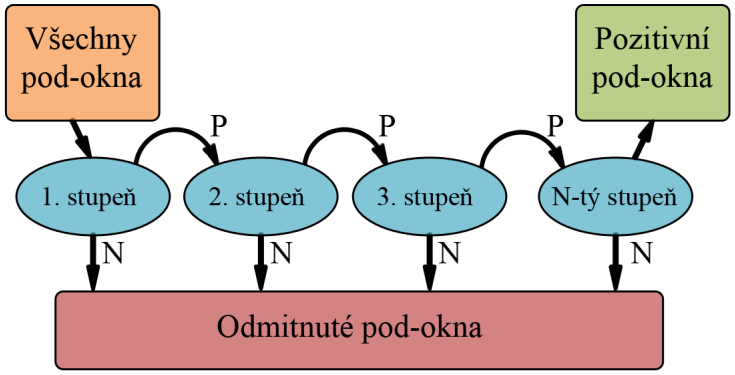


Obrázek 14: Nelineárně oddělitelné trénovací data.

3.2 Kaskáda klasifikátorů

Algoritmus pro konstrukci kaskády klasifikátorů, který zajistí zvýšenou výkonnost detekce a zároveň výrazně snižuje výpočetní čas. Klíčovou myšlenkou je, že menší, a tedy účinnější, posílené klasifikátory mohou být zkonstruovány tak, aby odmítly mnoho negativních pod-oken při detekci, téměř všechny pozitivní případy (tj. prahová hodnota posíleného klasifikátoru může být nastavena tak, aby se počet falešně negativních výsledků blížil nule). Jednodušší klasifikátory se používají k odmítnutí většiny pod-oken před složitějšími klasifikátory, které jsou vyvolány po dosažení nízkých falešně pozitivních hodnot. Celková podoba procesu detekce je degenerovaný rozhodovací strom, který nazýváme "kaskáda". Ukázka na obrázku [15]. Pozitivní výsledek z prvního klasifikátoru spouští vyhodnocení druhého klasifikátoru, který byl také upraven pro dosažení velmi vysoké míry detekce. Pozitivní výsledek z druhého klasifikátoru spouští třetí klasifikátor, a tak dále. Negativní výsledek v libovolném místě vede k okamžitému odmítnutí pod-okna.

Fáze v kaskádě jsou konstruovány pro trénování klasifikátorů pomocí metody Adaboost a poté nastavení prahové hodnoty, k minimalizaci chybně negativních hodnot. Výchozí práh AdaBoost je navržen tak, aby získal nízkou míru chyb v trénovacích datech. Obecně nižší práh přináší vyšší míru detekce a vyšší počet falešně pozitivních hodnot.



Obrázek 15: Kaskáda klasifikátorů.

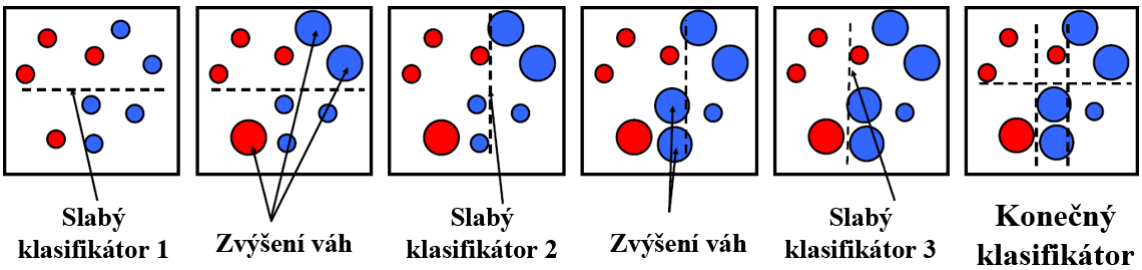
Řada klasifikátorů je aplikována na každé pod-okno. Počáteční klasifikátor eliminuje velké množství negativních příkladů s velmi malým zpracováním. Následné vrstvy odstraní další negativa, ale vyžadují další výpočty. Po mnoha zpracovaných fází počet pod-oken bylo postupně sníženo. Další zpracování může mít jakoukoli formu, jako jsou dodatečné fáze kaskády, nebo alternativní systém detekce.[2]

3.3 Adaptive Boost

AdaBoost je nejvýznamnější zástupce algoritmů založených na principu boostingu. Metoda, která vytváří vysoce přesný nelineární klasifikátor kombinací zvyšováním mnoho relativně slabých a nepřesných lineárních klasifikátorů.[11] Ukázka na obrázku[16]. AdaBoost je použitelný s širokým spektrem příznaků a není náchylný k přetrénování jako neuronové sítě. Působí jako meta-algoritmus. Takovýto výsledný klasifikátor je daný vzorcem[12]:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{3.7}$$

kde $h_t(x)$ jsou slabé klasifikátory a $H(x)$ je silný klasifikátor. Trénovací data pro tento algoritmus jsou definována jako uspořádaná dvojce učebního vzoru a zařazená do klasifikační třídy. Formální definice může vypadat například následovně: $(x_1, y_1), ..., (x_m, y_m)$, $x_i \in X, y_i \in Y$. Klasifikátor je matematický stroj, který vzoru ze vstupní množiny přiřazuje prvek z výstupní množiny $h : X \to Y$. Pokud je uvažována binární kombinace, obsahuje výstupní množina pouze dva prvky. $Y = \{-1, +1\}$



Obrázek 16: Výsledný klasifikátor vytvořený kombinací slabých klasifikátorů.

3.3.1 Slabý klasifikátor

Slabý klasifikátor bývá často reprezentován jednoduše a rychle získatelnou hodnotou. Příkladem může být prahová hodnota příznaku, ale dá se použit i výstup neuronové sítě, nebo rozhodovací strom. Množství výběru klasifikátoru jsou rozsáhle. Jediným omezujícím kriteriem je, aby chyba slabého klasifikátoru byla na trénovací množině menší než 50%. Nechť $\{D_i, i = 1, ..., N\}$ je tréninkový rozdíl příkladové sady, kde N je počet tréninkových vzorků. Každý příklad je spojen s označením $\{l_i, i = 1, ..., N\}$ a $l_i = 1$ v případě, že výňatek se zjišťuje z příchozích snímků s jednou osobou, jinak $l_i = 0$. Každý rozdílný příklad i je reprezentován jako d-rozměrný funkční vektor, který se skládá z nějaké globální a lokální informace. Váhy, $\frac{1}{2N_p}, \frac{1}{2N_n}$, jsou zprvu nastaveny k trénování pozitivních a negativních vzorků, resp. kde N_p je počet pozitivních vzorků, a N_n je počet negativních vzorků. Obecná forma slabého klasifikátoru AdaBoostu je diskriminantní funkce, která

je definována jako[12]:

$$h(D_i) = \begin{cases} 1 & h^T \cdot D_i \geq 0 \\ -1 & \text{jinak} \end{cases} \tag{3.8}$$

kde h představuje diskriminační nadrovinu trénovanou způsobem nejmenších čtverců.

Časová koherence objevených osob je využívána k udržování seznamu T klasifikátorů, které jsou natrénované v průběhu času. Jakmile je příchozí slabý klasifikátor detekován, a následně posouzen jako objevená osoba, vyřadí se nejstarší slabý klasifikátor a natrénuje se nový slabý klasifikátor na datech přidaných z nejnověji dostupných příkladů, a rekonstruuje se silný slabý klasifikátor. To znamená, že osoba v další časové periodě bude brána jako objevená a nebude znovu objevována. Specifická osoba, která bude objevená, může být začleněná a nastavená do množiny trénovacích dat jako jeden, nebo více příkladů, které se podílejí na trénování slabého klasifikátoru, ale nemůže být odstraněna ve fázi výsledné aktualizace pro silný klasifikátor. Konkrétní algoritmus je dán: (Vypis [1])

Vstup : Vstupní sekvence chodců $\{P1, ..., P_m\}$
Předem stanovený klasifikátor $C(x)$
Výstup : Výsledek sekvencí znovu detekovaných chodců $\{r1, ..., r_n\}$

Inicializace (Opakujte v 1..)

1. Shromažďování souboru dat $\{D_i\}_{i=1}^N, \{l_i\}_{i=1}^N$.
2. Inicializace vah $\{w_i\}_{i=1}^N za \frac{1}{2N_p}, \frac{1}{2N_n}$.
3. For $t = 1, ..., T$
   (a) Trénování slabého klasifikátoru h_t
   (b) Nastavení $err = \sum_{i=1}^N w_i |h_t(D_i) - l_i|$
   (c) Nastavení váhy slabého klasifikátoru $\alpha_t = \frac{1}{2} \log \frac{1-err}{err}$
   (d) Aktualizace vah příkladu $w_i = w_i e^{(\alpha_t |h_t(D_t)^{-l_t})|}$
4. Silný klasifik átor značený $H(x)$ je dán: $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$
5. Opakujte, dokud H_i nedosáhne lepších výsledku, než předem stanovený klasifikátor.

Foreach nový vložený chodec P_j dělej:

1. Výňatek $\{D_i\}_{i=1}^{N_j}$ příkladů
2. Testování příkladů použitím silného klasifik átoru $H(x)$,označený jako $\{l_i\}_{i=1}^{N_j}$
3. Výstup $r_j \in [0, ..., j-1]$, 0 nepředstavuje shodu
4. Odstranění K nejstarší ze slabých klasifikátor
5. Revidování váhy $w_i = w_i(1 - \frac{N_j}{\sum_{k=1}^j N_k}), i \in |1, \sum_{k=1}^{j-1} N_k|$
6. Inicializace vah $w_i = 1 - \frac{N_j}{\sum_{k=1}^j N_k}$ pro nové příklady
7. For $l = K + 1...T$, (Aktualizace vah)
   (a) Výběr $h_t(x)$ s minimální chybou err
   (b) Aktualizace α_ta w_i, potom odstranění $h_t(x)$ z $\{h_{K+1}(x), ..., h_T(x)\}$
8. For $t = 1...K$,(Přidaní nových slabých klasifikátorů)
   (a) Trénování slabých klasifik átorů h_t
   (b) Počítáni err a α_t, Aktualizace vah $\{w_i\}$
9. Aktualizovaný silný klasifik átor značený $H(x)$ je dán: $H(x) = \sum_{t=1}^T \alpha_t h_t(x)$

Výpis 1: AdaBoost algoritmus - Funkce Silného klasifikátoru.[12]

3.3.2 Aktualizace funkce pro silný klasifikátor

V aktualizačním stavu, algoritmus odstraní K staré slabé klasifikátory k vytvoření prostoru pro K nových slabých klasifikátorů. Nicméně, před přidáním nového slabého klasifikátoru je potřeba aktualizovat váhu zbývajících slabých klasifikátorů. Krok (7) z algoritmu: (Vypis [1]), aktualizační stav aktualizuje váhy zbývajícího slabého klasifikátoru, místo trénování nového slabého klasifikátoru na celé množině trénovacích dat. Slabý klasifikátor trénuje jednoduše dáním starých příkladů a nového rozdělení založeno na původních vahách, takže není třeba váhy vypočítávat znovu, jako v kroku (5). To šetří čas při trénování a vytváří silný klasifikátor, stejně jako vzor rozdělení, které mohou být použity pro trénink nového slabého klasifikátoru, jako se provádí v kroku (8).[12]

Původní verze Adaboost prováděla pouze binární klasifikaci, v současnosti existují modifikované varianty umožnující klasifikaci reálným výstupem. Reálný výstup vykazuje míru příslušnosti do dané třídy, tato vlastnost vede většinou ke konstrukci přesnějších klasifikátorů. Různé varianty boostingu jsou známe jako Discrete AdaBoost, Real AdaBoost LogitBoost, a Gentle Adaboost, který byl použitý v této práci. Všechny z nich jsou velmi podobné ve své celkové struktuře.[10]

4 Procesy trénování a detekce

4.1 Proces trénování metod LBP a Haar s použitím AdaBoostu

Proces trénování se skládá ze tří kroků:

1. Obrázková akvizice
2. Vytváření vzorků
3. Trénování

4.1.1 Obrázková akvizice

Získávání snímků lze odkudkoli, kde si můžeme najít chodce pomoci kamerového systému, digitálního fotoaparátu nebo videa. Sbírka datového souboru se skládá ze sbírky pozitivních vzorků a sbírky negativních vzorků. Obrázek [17] je ukázkou těchto vzorků. Pozitivní a negativní vzorky jsou z důvodu jednoduchosti umístěné v různých složkách. Trénování detektoru pro chodce není jednoduché, protože chodci mění svojí pózu. Proto je lepší získat vzorky v nejrůznějších pózách.

Obrázek 17: Nahoře pozitivní a dole negativní vzorky.[18][19]

4.1.2 Vytváření vzorků

Za účelem dobré funkce má OpenCV vytvořený program k trénování klasifikátoru. Nezbytné k trénování klasifikátoru je použití pozitivních vzorků, které byly získané během prvního kroku, ale nemůžeme přivádět pozitivní snímky přímo do trénovacího programu, protože program jen uznává číselné hodnoty, které určují umístění chodce v obrázku. Pro získání těchto potřebných hodnot (x-souřadnice, y-souřadnice, šířka, výška)

má OpenCV program **opencv_createsamples.exe**, který slouží k vytváření vzorků. Tento program z pozitivních snímků vytvoří vzorky do vektorového souboru typu **.vec**, např. Pos.vec. Tento vektorový soubor bude použitý při trénování klasifikátoru.

Použití opencv_createsamples pro vytvoření vzorků[13]:

Pro spuštění programu opencv_cratesamples se používá příkazový řádek, pomocí příkazového řádku se přejde na místo, kde je OpenCV nainstalován, do složky kde je umístěný **.exe** soubor programu. Následně se použije příkaz s parametry pro spuštění. Ukázka příkazu:

opencv_createsamples.exe -info pos.txt -vec samples.vec -w 24 -h 48 -num 23000

Parametry:

- **-info** <kolekce souborů>, soubor obsahující umístění obrázků chodců ukázka na obrázku [18]
- **-vec** <název vektorového souboru>, počet pozitivních vzorků k trénování
- **-w** <šířka vzorku>, šířka obrázku výsledných vzorků používané pro trénování
- **-h** <výška vzorku>, výsledná výška obrázku
- **-num** <počet vzorků>, binární soubor v souladu s počtem vzorků

pos/pos1/pos1_1.png 1 0 0 24 48
pos/pos1/pos1_2.png 1 0 0 24 48
pos/pos1/pos1_3.png 1 0 0 24 48
pos/pos1/pos1_4.png 1 0 0 24 48
pos/pos1/pos1_5.png 1 0 0 24 48
pos/pos1/pos1_6.png 1 0 0 24 48
pos/pos1/pos1_7.png 1 0 0 24 48
pos/pos1/pos1_8.png 1 0 0 24 48
pos/pos1/pos1_9.png 1 0 0 24 48
pos/pos1/pos1_10.png 1 0 0 24 48
pos/pos1/pos1_11.png 1 0 0 24 48
pos/pos1/pos1_12.png 1 0 0 24 48
pos/pos1/pos1_13.png 1 0 0 24 48
...

Obrázek 18: Ukázka souboru s umístěním pozitivních vzorků.

4.1.3 Trénování

Předchozími kroky jsme v našem případě, pro proces trénování klasifikátoru rozpoznávajícího postavy, použili sbírku dat, která obsahuje 2300 pozitivních obrázků postavy o rozměrech 24×48 pixelů (šířka a výška)[18][17][19]. Popředí je tvořeno postavou z frontálního pohledu. Pozadí obsahuje artefakty netypické pro postavy. Sada pro trénink byla kvůli usnadnění výpočtu převedena do šedotónové reprezentace. Sada negativních obrázků obsahující prostředí města, přírody, atd. bez přítomnosti osob. Tato sada obsahuje 11000 negativních obrázků také o rozměrech 24×48, které jsou použity pro samostatné trénování. Jak už bylo zmíněno, tak OpenCV má pro trénování klasifikátoru zabudované programy. Jedním z nich byl opencv_createsamples, tím dalším je **opencv_traincascade.exe**. Tento program je použit pro trénování kaskád klasifikátoru. Trénování použitím OpenCV je založená na přístupu AdaBoostu. Hlavním cílem AdaBoost algoritmu je natrénovat silný klasifikátor lineární kombinací silných vlastnosti slabého klasifikátoru a vlastnosti, které představují postavu (chodce) z trénovací sady. Ukázka na obrázku [19].

Obrázek 19: Trénování s AdaBoostem.

Použití opencv_traincascade pro trénování klasifikátorů[13]:

Program opencv_traincascade se stejně jako program pro vytváření vzorků spouští příkazovým řádkem.
Ukázka příkazu:

opencv_traincascade.exe -data classifier_lbp_03 -vec samples.vec -bg neg.txt -numStages 25 -minHitRate 0.999 -maxFalseAlarmRate 0.5 -bt GAB -numPos 2300 -numNeg 11000 -w 24 -h 48 -mode ALL -featureType LBP > lbp_03.txt

Parametry:

- **-vec** <název vektorového souboru>, soubor s pozitivními vzorky vytvořené nástrojem opencv_createsamples
- **-bg** <název souboru>, soubor s umístěním negativních vzorků
- **-numStages** <počet etap>, počet kaskádových etap
- **-minHitRate** <minimání HitRate>, minimální požadována hodnota míry pro všechny etapy klasifikátoru
- **-maxFalseAlarmRate** <maximální FalseAlarmRate>, maximální požadována hodnota falešných poplachů pro všechny etapy klasifikátorů
- **-bt** <{DAB, RAB, LB, GAB}>, Typ posílených klasifikátorů: GAB -Gentle AdaBoost

- **-numPos** <počet pozitivních vzorků>,
- **-numNeg** <počet negativních vzorků>, počet pozitivních/negativních vzorků v oblasti trénováni pro všechny etapy klasifikátoru
- **-w** <šířka vzorku>,
- **-h** <výška vzorku>, Velikosti vzorků odborné přípravy (v pixelech), které musí mít přesné hodnoty jako při vytváření vzorků pomoci nástroje opencv_createsamples
- **-mode** <BASIC, CORE, ALL>, Basic je výchozí nastavení, které obsluhuje pouze svisle prvky, zatímco ALL využívá úplnou sadu svislé prvky a pootočené prvky o 45 stupňů
- **-featureType** <typ funkce>, Haar a LBP
- **>** <název souboru>, textový soubor s informacemi o průběhu trénování

Jakmile dokončí opencv_traincascade program svou práci, je natrénovaná kaskáda uložená do **cascade.xml** souboru ve složce, který byl zadán jako -data parametr. Ostatní soubory v této složce jsou vytvořeny pro případ přerušení tréninku, kde program začne od naposledy vytvořené etapy. Z textových souboru, který jsme dostali posledním parametrem, jsme zjistili, že metoda LBP natrénovala klasifikátor s 256 příznaky a metoda Haar s počtem 687 příznaků.

4.2 Proces trénování metody HOG s použitím SVM klasifikátoru

V této části jsme nepoužili, žádné vytvořené aplikace k trénování klasifikátoru z knihovny OpenCV, ale detektor jsme natrénovali za použití programu Microsoft Visual Studio 2012 s pomocí knihovny OpenCV. Pro trénování, tak jako u procesu trénování metod LBP a Haar, i tady je zapotřebí obrázková akvizice. Byla použitá stejná sada trénovacích snímku a to 2300 pozitivních a 11000 negativních. U těchto snímků bylo potřeba změnit velikost na 64×128 pixelů. HOG deskriptor byl nastaven na velikost okna 64×128, velikost bloků 16×16 a jejich krok 8×8. Velikost buněk 8×8, které jsou rozdělený na 9 kanálů. Pro tyto parametry bylo nastaveno počet příznaků, získaného z detekčního okna, na 3780. SVM je použit z knihovny OpenCV. Parametr C je nastaven na hodnotu 0,01, kde tento parametr reguluje velikost pásma a množství klasifikačních chyb. Tato hodnota byla nastavena z informací dosažené z článku [1], kde autoři touto hodnotou dosáhli nejlepších výsledků. Typ jádra (kernel type) je typu linear.

4.3 Proces detekování chodců

Pro aplikování natrénovaných klasifikátorů byl v této práci použit software Microsoft Visual Studio 2012 s knihovnou OpenCV. Prvním krokem procesu, je načtení snímku, ve kterém bude detektor hledat chodce. Dalším krokem je načtení natrénovaných kaskád klasifikátorů. Následuje krok, kdy je převeden snímek do šedotónové reprezentace, ale před tímto krokem, lze v případě potřeby aplikovat vyhlazovací filtr. Poté je na snímek aplikován klasifikátor pro detekci chodců. Posledními kroky je ohraničení detekovaných chodců obdélníkem a uložení obrázku do určeného adresáře.
Ukázka příkazu pro aplikování klasifikátoru [15]:
pedestrian_cascade.detectMultiScale(frame_gray, pedestrians, 1.10, 2, 0|CASCADE_SCALE_IMAGE, cv::Size(8,8), cv::Size(640,800));
Parametry:

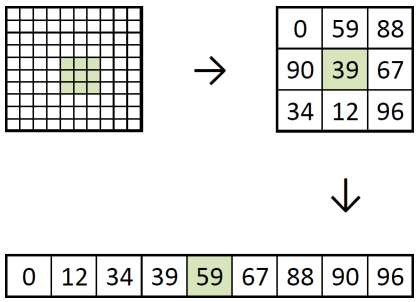
- pedestrian_cascade, **cascade** - načtená kaskáda ze souboru pomocí příkazu **Load()**
- frame_gray, **image** - matice obsahující snímek, kde jsou zjišťovány objekty
- pedestrians, **objects** - vektor obdélníků, kde každý obdélník obsahuje detekovaný objekte
- 1.10, **scaleFactor** - parametr určující, jak moc je obrazová velikost snižována u každého obrazového měřítku.
- 2, **minNeighbors** - parametr určující, kolik sousedících kandidátů na obdélník musí zachovat
- 0|CASCADE_SCALE_IMAGE, **flags** - režim provozu, který má čtyřmi platné nastavení kombinované s operátorem OR, CASCADE_SCALE_IMAGE je jedním nastavením a v algoritmu se používá pro škálování obrazu, který přináší výkonnostní výhody pro pomět a vyrovnávací paměť
- cv::Size(8,8), **minSize** - minimální možná velikost objektu. Menší objekty jsou ignorovány
- cv::Size(640,800), **maxSize** - maximální možná velikost objektu. Větší objekty jsou ignorovány

4.3.1 Vyhlazování

Při použití metod pro detekci v obraze je potřeba řešit problémy s citlivostí na šum, tato citlivost vede k detekování nepřehlédnutelného počtu neexistujících hran i při malém zašumění v obraze. Tyto problémy je možné do jisté míry snížit pomocí metod vyhlazování, nebo také rozmazání obrázků, která je aplikována na obraz před tím, než je započat proces detekování. Existuje mnoho různých druhů filtrů, v této práci jsme použili pouze dva filtry, Gaussián filtr a Median filtr, které si následně popíšeme.[14]

4.3.1.1 Medián filtr Tento filtr projde každý pixel obrazu a nahradí každý pixel s mediánem ze sousedních pixelů (umístěné ve čtvercovém sousedství kolem hodnoceného pixel). Ukázka určení mediánu je na obrázku [20]. Tento filtr zajišťuje funkce *medianBlur(src, dst, i)*, který používá tři argumenty:

- **src** - zdrojový obrázek
- **dst** - místo určení, musí být stejného typu jako src
- **i** - velikost jádra, hodnota musí být lichá a pouze jedna, protože používá čtvercová okna



Obrázek 20: Medián setříděných hodnot.

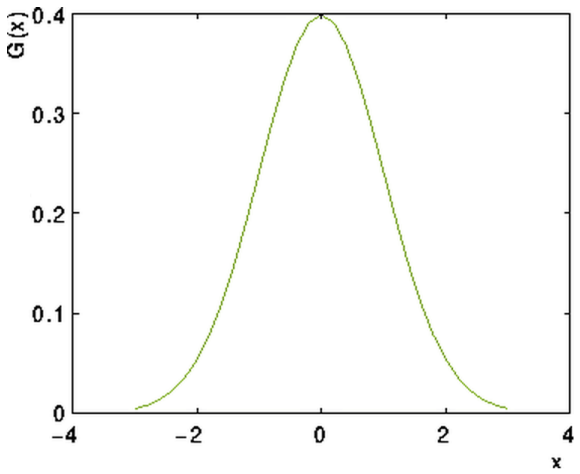
4.3.1.2 Gaussián filtr Pravděpodobně nejvíce užitečný filtr (i když ne nejrychlejší). Gaussián filtr provádí konvoluci každého bodu do vstupního pole s Gaussiánovým jádrem a následně všechno sečte k vytvoření výstupního pole. Ukázka tohoto filtru je na obrázku [22]a a na obrázku [22]b je Medián filtr. Pro jasnější obraz si stačí vzpomenout jak vypadá 1D Gaussiánovo jádro. Ukázka na obrázku [21]. Za předpokladu, že obraz je 1D, tak pixel, který se nachází ve středu bude mít největší váhu. Váha jeho sousedů snižuje prostorovou vzdálenost mezi nimi a zvyšuje středový pixel. 2D Gaussián je reprezentován jako[14]:

$$G_0(x, y) = Ae\frac{-(x-\mu_x)^2}{2\sigma_x^2} + \frac{-(y-\mu_y)^2}{2\sigma_y^2} \tag{4.1}$$

Tento filtr zajišťuje funkce *GaussianBlur (src, dst, Size(i ,i), 0, 0)*, který používá tyto argumenty:

- **src** - zdrojový obrázek
- **dst** - místo určení
- **Size(w, h)** - Velikost jádra, kde **w** a **h** musí být kladná a lichá čísla. Jinak velikost bude počítána pomocí σ_x a σ_y.

- σ_x - standardní odchylka x, psáni **0** znamená, že σ_x se počítá s použitím velikosti jádra
- σ_y - standardní odchylka y, psáni **0** znamená, že σ_y se počítá s použitím velikosti jádra

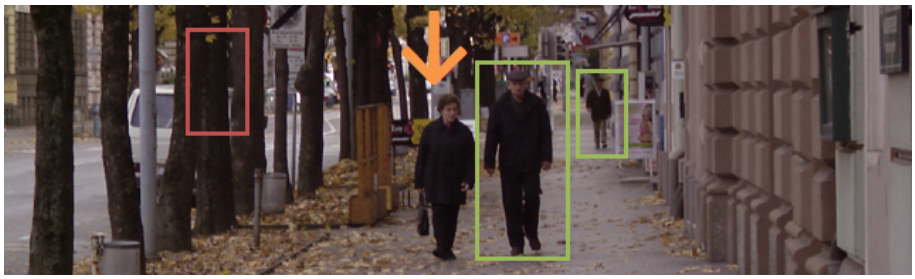


Obrázek 21: Gaussiánovo 1D jádro.[14]

Obrázek 22: Rozdíl Gaussián filtru a Medián filtru.

4.4 Hodnocení kvality modelů

Výstupem celého testovacího procesu je zjištění v kolika případech klasifikátor data zařadil do správné třídy a kolikrát selhal. Správně pozitivní klasifikace je značena jako TP a obdobně správně negativní je značena TN. Chybně pozitivní klasifikace je označena FP a chybně negativní klasifikace je FN. Ukázka těchto klasifikací je na obrázku [23], kde zelené obdélníky jsou TP klasifikace, a tedy správně klasifikovali chodce. Oranžová šipka ukazuje na místo, kde se vyskytuje chodec, ale toto místo nebylo správně klasifikováno, a proto klasifikace tohoto místa je označeno jako FN. Červený obdélník je FP klasifikace, který v části fotografie klasifikoval chodce, i když reálně tam není. Tato FP klasifikace je vykreslená, protože v blízkém okolí jsou takovéto chybné klasifikace, nebo také obdélníky, aspoň dva, a tedy splňuje nastavenou hodnotu parametru *minNeighbors* u příkazu *detectMultiScale*, který je vysvětlený v předchozí kapitole. Všechny ostatní části obrázku jsou klasifikována jako TN, takovýchto části je obrovské množství, a pro naší práci experimentálního porovnání jsou tyto části nepotřebné. Z právě zmíněných hodnot klasifikací lze vypočítat charakteristiky, které dávají jasnější představu o chování modelu. Tyto parametry jsou blíže představeny v následujících řádcích[16]:



Obrázek 23: Ukázka klasifikace.

Citlivost (angl. Sensitivity)

Citlivost, také známá jako správně pozitivní míra, je měřítkem toho, jak dobře je klasifikátor schopný správně předpovědět skutečné pozitivní vzorky. Jeho formulace je:

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.2}$$

Přesnost (angl. Precision)

Přesnost, také známa jako pozitivně předpokládaná hodnota, což je míra správně pozitivní hodnoty s ohledem na všechny předpokládané pozitiva a F1-skóre. Formulace přesnosti je:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

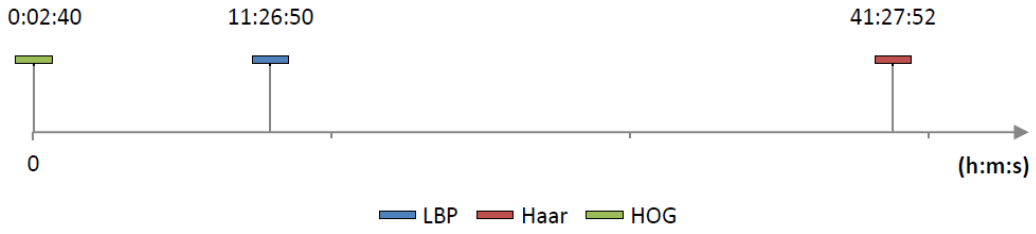
F1-Score

F1-skóre, nebo také F-míra, je všeobecné opatření přesnosti klasifikátoru, který kombinuje přesnost a citlivost. Formulace F1-Score je:

$$F1\text{-}Score = 2 \cdot \frac{precision \cdot sensitivity}{precision + sensitivity} = 2 \cdot \frac{TP}{(2 \cdot TP + FP + FN)} \tag{4.4}$$

5 Experimety

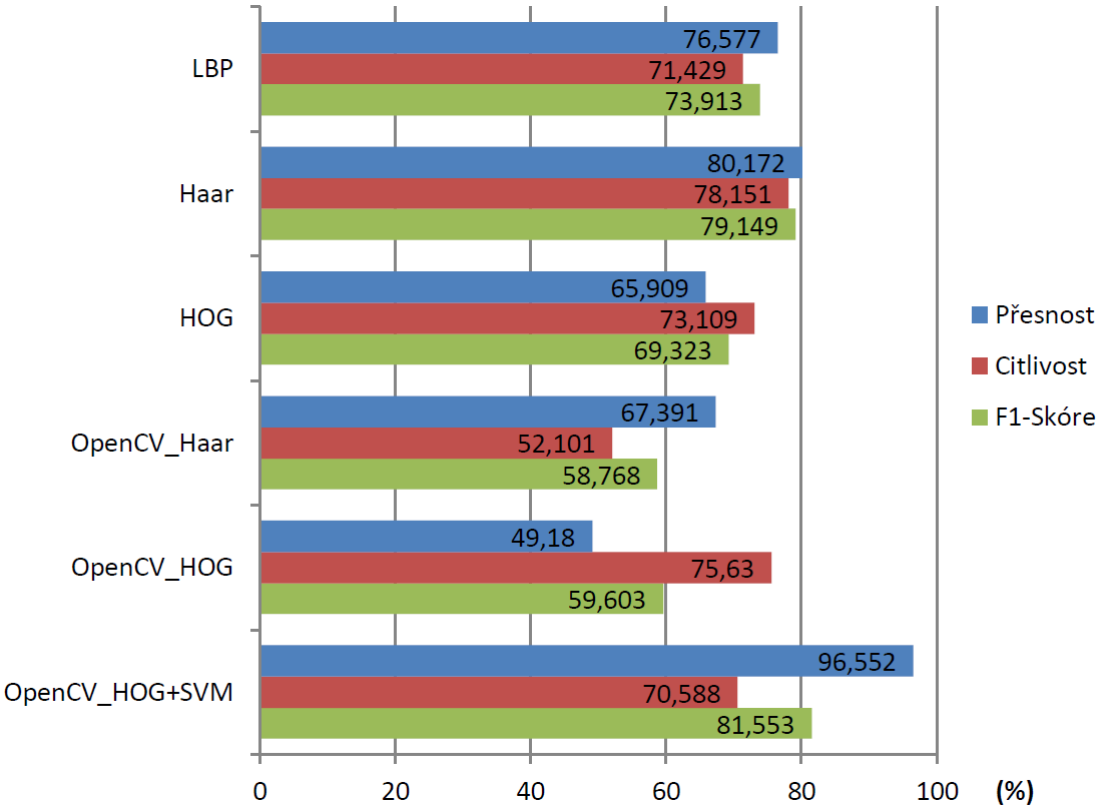
Všechny experimenty, byly prováděny na sadě testovacích datech, která obsahovala 60 snímků. Tyto snímky se liší ve velikosti okna i počtem chodců. Pro experimenty byly natrénované metody pro příznakové rozpoznávání. Mezi těmito metody jsou renomované příznaky HOG, LBP a Haar. Dále pro porovnání byly použité metody, které byly natrénované v rámci knihovny OpenCV. Mezi tyto metody patří kaskády HOG (OpenCV_HOG) a Haar (OpenCV_Haar), a metoda HOG s SVM klasifikací (OpenCV_HOG+SVM). U těchto metod neznáme trénovací sadu pro natrénováni, nebo čas trénování. Doba trénování metod pro náš experiment je znázorněna na obrázku [24]. Metody Haar a LBP byly natrénovány na školním zařízeni s procesorem Xeon obsahující 16 fyzických jader. Metoda HOG byla natrénována na osobním notebooku HP ProBook 4530s s procesorem Intel(R) Core(TM) i5-2430M CPU 2.40GHz. Tato metoda byla na osobním notebooku, oproti metodám trénovaných na výkonném zařízení, natrénována velmi rychle. V následující části jsou popsány experimenty, které byli rozděleny na základě použití vyhlazení.



Obrázek 24: Graf znázorňující dobu trénování metod

5.1 Použití příznakových metod

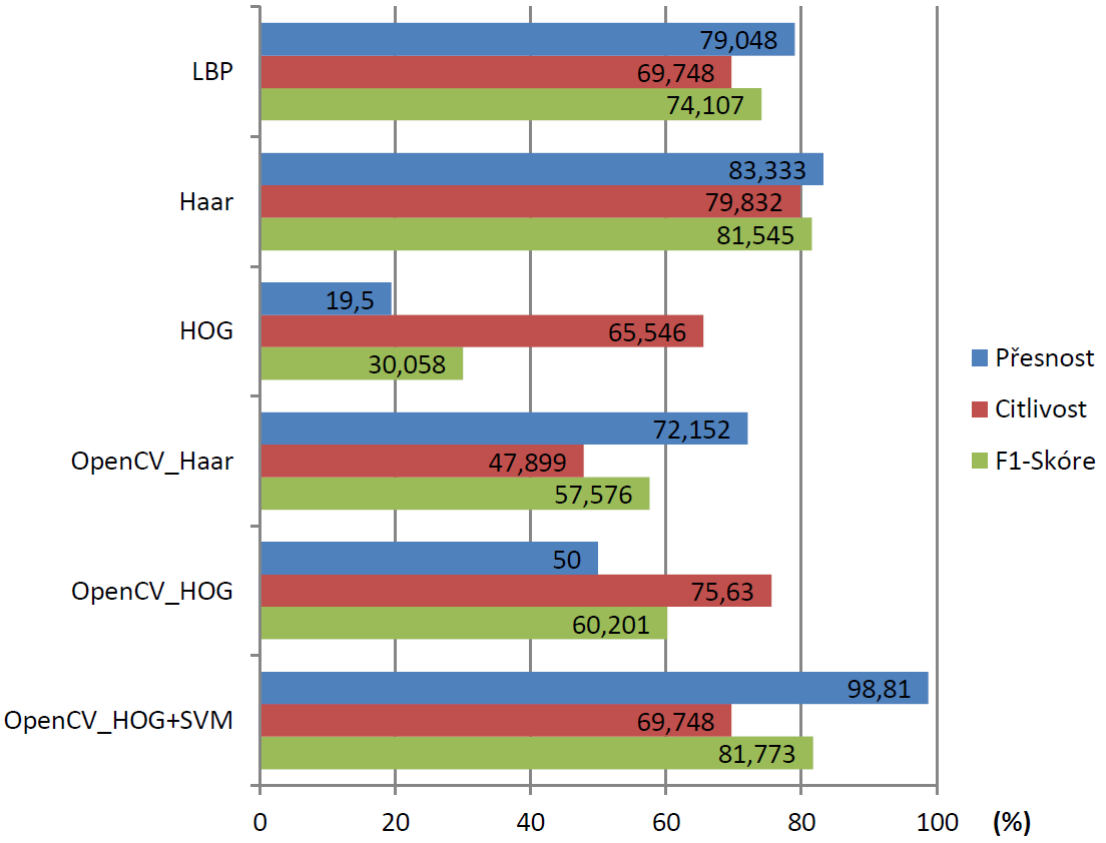
Prvním experimentem pro detekci osob bylo použití příznakových metod bez použití vyhlazení. Získaná data jsou v tabulce [1] v příloze [B]. Na základě těchto dat byl vytvořen graf, který je zobrazen níže na obrázku [25]. Z údajů lze vyhodnotit, že metoda OpenCV_HOG+SVM je znatelně přesnější než ostatní metody, a to s přesností 81,553% F1-Skóre (Míra přesnosti). Naproti tomu metoda OpenCV_Haar s 58,768% je nejméně přesná. Detekční metody, které byly natrénovány jsou srovnatelné. Tyto metody se liší nepatrně, i když se dá říct, že metoda Haar je přesnější. Metody z knihovny OpenCV jsou od sebe velmi odlišné, a vůči natrénovaným metodám jsou méně přesné, s výjimkou nejpřesnější metody OpenCV_HOG+SVM. Ukázka, jednotlivých příznakových metod pro detekování objektů, je v příloze [C.1] na obrázku [29].



Obrázek 25: Graf s charakteristikami pro porovnání metod.

5.2 Použití příznakových metod s Medián filtrem

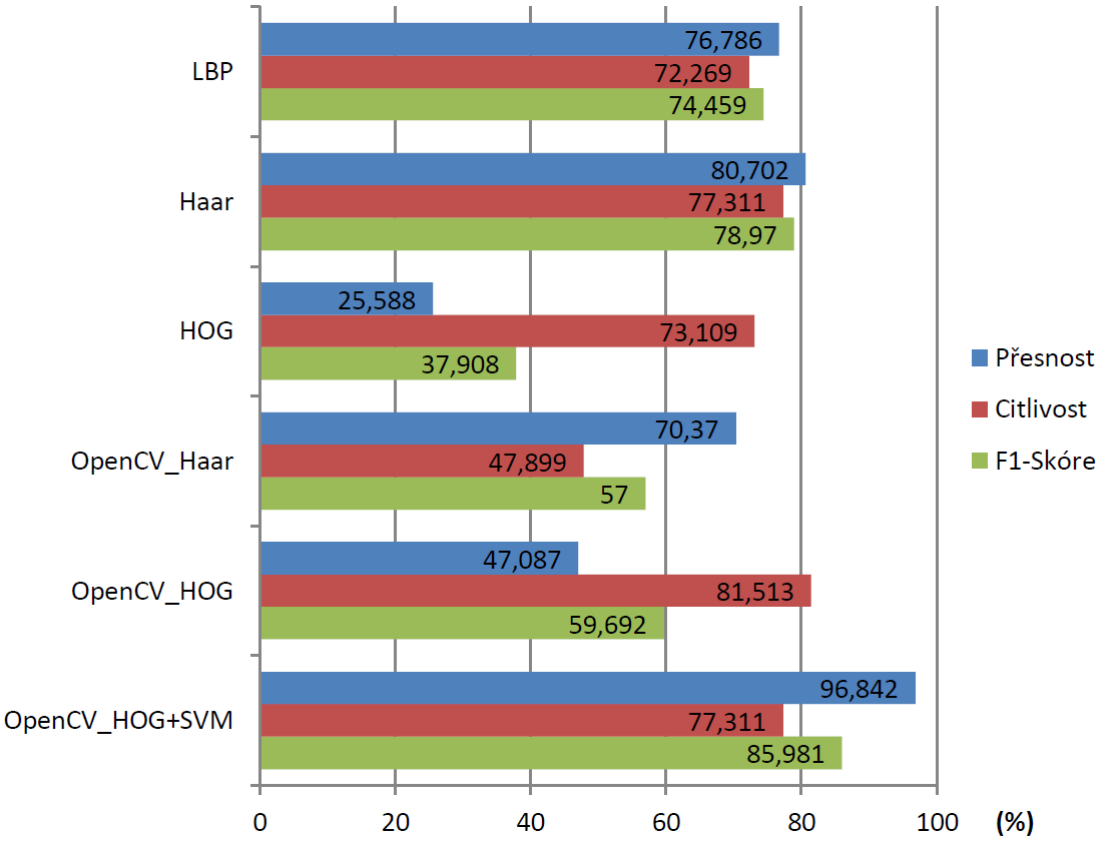
Dalším experimentem bylo použití příznakových metod s použitím vyhlazení. Prvním vyhlazovacím filtrem byl použit Medián filtr. Z tabulky [2] v příloze [B] byl vytvořen graf [26]. Z tohoto grafu lze vyhodnotit, že u metod LBP a Haar je přesnost, oproti experimentu bez vyhlazení, nepatrně zvýšená. Také u metod z knihovny OpenCV, a to OpenCV_HOG a OpenCV_HOG+SVM, je přesnost nepatrně zvýšená, ale OpenCV_Haar metoda je nižší. Nejvýraznější změnu má natrénována metoda HOG, která je snížena oproti experimentu bez vyhlazení na 30,058%. Příčina této extrémní sníženosti přesnosti je zvýšen počet chybně pozitivních hodnot, které bylo způsobeno použitím vyhlazení. V tomto případě vyhlazení snížilo počet hran, které klasifikátor následně špatně vyhodnotil a vyskytly se chybně pozitivní hodnoty tam kde nejsou. Ukázka, jednotlivých příznakových metod s použitím Medián filtru, je v příloze [C.2] na obrázku [30].



Obrázek 26: Graf s charakteristikami pro porovnání metod s použitím Medián filtru.

5.3 Použití příznakových metod s Gaussián filtrem

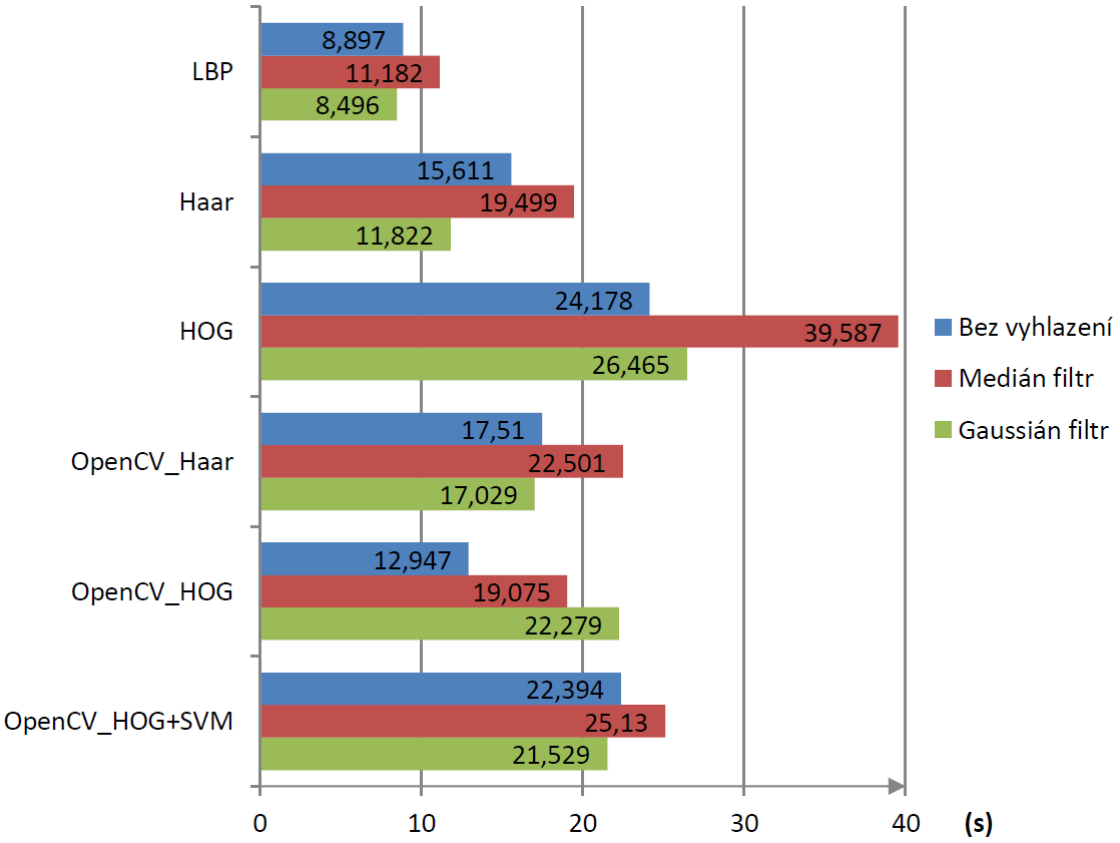
Další použití vyhlazení a posledním experimentem bylo použití příznakových metod s použitím Gaussián filtru. Z tabulky [3] v příloze [B] byl vytvořen graf [27]. Z tohoto grafu lze vyhodnotit, že metoda HOG, která vůči použití Medián filtru, je o něco přesnější, ale stále je přesnost extrémně snížena, a to na 37,908%. Přesnost metod LBP, OpenCV_Haar a OpenCV_HOG+SVM, v případě srovnání s prvním experimentem bez vyhlazení, je zvýšená. Metoda OpenCV_HOG+SVM s Gaussián filtrem je nejpřesnější ze všech experimentovaných metod. Zbývající metody Haar a OpenCV_HOG, mají přesnost nepatrně sníženou. Ukázka, jednotlivých příznakových metod s použitím Gaussián filtru, je v příloze [C.3] na obrázku [31].



Obrázek 27: Graf s charakteristikami pro porovnání metod s použitím Gaussián filtru.

5.4 Čas detekce

Na základě údajů všech tabulek v příloze [B] je zřejmé, že metody s využitím Medián filtru jsou pomalejší. Metoda s nejpomalejší detekci je HOG s použití Medián filtru. Tato metoda trvala 39,587 vteřin. Nejrychlejší detekční metoda je LBP s použitím Gaussián filtru, která trvala 8,496 vteřin. Tato metoda je nejrychlejší s použitím i bez použití jakéhokoliv filtru. Měřený čas detekce byl prováděn na celé testovací sadě o 60 snímcích.



Obrázek 28: Graf s detekčními časy.

6 Závěr

Práce se zabývala detekci chodců v obrazech. Cílem práce bylo seznámit se s metodami příznakového rozpoznávání objektů pomocí renomovaných příznaků. Vybranými metodami byly histogramy orientovaných gradientů s klasifikačním algoritmem Support Vector Machine. Další metody byly Haarovy příznaky a lokální binární vzory s algoritmem AdaBoost. S pomocí knihovny OpenCV tyto detektory měly být vytvořeny a měla být ověřená funkčnost, přesnost a rychlost těchto detektorů.

V práci bylo úspěšně dosaženo natrénování příznakových metod z renomovaných příznaků pomocí vytvořené trénovací sady. Tyto natrénované metody byly pro zajímavost ověřený spolu s předem natrénovanými metodami z knihovny OpenCV. V prostředí C++ byly tyto detektory implementovány. Metody, bez použití vyhlazení, dosahovaly velmi příznivých výsledků, kde tyto metody byly mnohem přesnější a mnohem rychlejší při detekci, než kaskády metod z knihovny OpenCV. Metoda HOG s klasifikací SVM z této knihovny, ale byla nejpřesnější. Metoda Haar byla z natrénovaných metod nejpřesnější a to s přesností přesahující 79%, ale čas detekce byl skoro dvojnásobný než metoda LBP, která dosahovala velmi dobrou přesnost skoro 74%. Také metoda HOG dosahovala slušných výsledků 69%. Ale tato metoda s použitím obou vyhlazovacích filtrů rapidně klesla pravděpodobně z důvodů, že použitím vyhlazení zaniklo mnoho hran a trénovací data nebyly před procesem trénování vyhlazeny. Klasifikátor tedy vyhodnotil příliš mnoho chybně pozitivních klasifikací. Použitím Medián filtru rychlost detekce velmi klesla, ale přesnost LBP a Haar metod se zvýšila. Gaussián filtr s těmito metody vykazuje nepatrně zanedbatelné změny v přesnosti a mírné zrychlení detekce.

V porovnání všech trénovaných metod se dá říct, že metoda LBP s použitím Gaussián filtru je v poměru přesnosti a rychlosti detekce nejlepší, včetně toho, že čas trénování metody je skoro čtyřikrát kratší. Ačkoli doba trénování metody HOG s klasifikací SVM je nesmírně krátká, tak doba detekce a přesnost jsou velmi nepříznivé. Tato práce lze do budoucna zdokonalit například rozšířením trénovací sady, přidáním jiných typů příznaků pro trénování detektoru, nebo vyhlazovacích filtrů.

7 Reference

[1] DALAL, Navneed; TRIGGS, Bill. *Histograms of oriented gradients for human detection* [online]. 2005
Dostupné z: http://ieeexplore.ieee.org/

[2] VIOLA, Paul; JONES, Michael. *Rapid Object Detection using a Boosted Cascade of Simple Features* [online]. 2001
Dostupné z: http://ieeexplore.ieee.org/

[3] MEENA, K.; SURULIANDI, A.. *Local Binary Patterns and its variants for face recognition* [online]. 2011
Dostupné z: http://ieeexplore.ieee.org/

[4] LIAO, Shengcai; ZHU, Xiangxin; ZHANG, Lun; LEI, Zhen; ZHANG, Lun; LI, Stan Z.. *Learning Multi-scale Block Local Binary Patterns for Face Recognition*, Berlin Heidelberg: Springer-Verlag, 2007. In: ICB pp. 828-837.

[5] GONZALES, Rafael C.; WOODS, Richard E.. *Digital Image Procesing. 3. vyd.*, New Jersey: Pearson Education, 2008. ISBN 978-0-13-168728-8.

[6] SOJKA, Eduard; GAURA, Jan; KRUMNIKL Michal. *Matematické základy digitálního zpracování obrazu* [online]. 2011
Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/digitalni_zpracovani_obrazu.pdf

[7] OpenCV. *Support Vector Machines. In: OpenCV API Reference* [online]. 2011-2014
Dostupné z: http://docs.opencv.org/modules/ml/doc/support_vector_machines.html

[8] GARCÍA, Fernando I.. *Introduction to Support Vector Machines In: OpenCV Tutorials* [online]. 2011-2014
Dostupné z: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[9] GARCÍA, Fernando I.. *Support Vector Machines for Non-Linearly Separable Data. In: OpenCV Tutorials* [online]. 2014
Dostupné z: http://docs.opencv.org/doc/tutorials/ml/non_linear_svms/non_linear_svms.html

[10] OpenCV. *Boosting. In: OpenCV API Reference* [online]. 2011-2014
Dostupné z: http://docs.opencv.org/modules/ml/doc/boosting.html

[11] SZEILESKI, Richard. *Computer Vision : Algorithms and Applications*, New York: Springer, 2011. 811 s. ISBN 978-1-84882-934-3.

[12] LEI, Hu; SHUQIANG, Jiang; QINGMING, HUANG; WEN, Gao. *People re-detection using Adaboost with sift and color correlogram* [online]. 2008
Dostupné z: http://ieeexplore.ieee.org/

[13] OpenCV. *Cascade Classifier Training. In: OpenCV Tutorials* [online]. 2011-2014
Dostupné z: http://docs.opencv.org/doc/user_guide/ug_traincascade.html

[14] OpenCV. *Smoothing Images. In: OpenCV Tutorials* [online]. 2011-2014
Dostupné z: http://docs.opencv.org/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html

[15] OpenCV. *Cascade Classification. In: OpenCV API Reference* [online]. 2011-2014
Dostupné z: http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html?highlight=detectmultiscale

[16] REYES-ORTIZ, Jorge L.. *Smartphone-Based Human Activity Recognition*, Springer Theses, 2015. ISBN 978-3-319-14274-6.

[17] ENWEILER, Markus; GAVRILA Dariu M.. *Daimler Pedestrian Detection Benchmark Dataset* [online]. 2009
Dostupné z: http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/Daimler_Mono_Ped__Detection_Be/daimler_mono_ped__detection_be.html

[18] DALAL, Navneed. *INRIA Person Dataset* [online]. 2005
Dostupné z: http://pascal.inrialpes.fr/data/human/

[19] MIT-CBCL. *CBCL Pedestrian Database* [online]. 2000
Dostupné z: http://cbcl.mit.edu/software-datasets/PedestrianData.html

A Obsah CD

Součástí přiloženého CD je:

- Elektronická verze této práce ve formátu PDF.
- Zdrojové kódy aplikace pro trénování metody HOG+SVM.
- Zdrojové kódy aplikace pro detekci osob.
- Natrénované klasifikátory.
- Sada trénovacích obrázků.
- Sada testovacích obrázků.
- Testovací obrázky po aplikovaní detekčních metod.

B Tabulky

Metody	TP	FP	FN	Přesnost (%)	Citlivost (%)	F1-Skóre (%)	Čas detekce (s)
LBP	85	26	34	76,577	71,429	73,913	8,897
Haar	93	23	26	80,172	78,151	79,149	15,611
HOG	87	45	32	65,909	73,109	69,323	24,178
OpenCV Haar	62	30	57	67,391	52,101	58,768	17,51
OpenCV HOG	90	93	29	49,18	75,63	59,603	12,947
OpenCV HOG+SVM	84	3	35	96,552	70,588	81,553	22,394

Tabulka 1: Tabulka metod a jejich charakteristik

Metody	TP	FP	FN	Přesnost (%)	Citlivost (%)	F1-Skóre (%)	Čas detekce (s)
LBP	83	22	36	79,048	69,748	74,107	11,182
Haar	95	19	24	83,333	79,832	81,545	19,499
HOG	78	322	41	19,5	65,546	30,058	39,587
OpenCV Haar	57	22	62	72,152	47,899	57,576	22,501
OpenCV HOG	90	90	29	50	75,63	60,201	19,075
OpenCV HOG+SVM	83	1	36	98,81	69,748	81,773	25,13

Tabulka 2: Tabulka metod a jejich charakteristik s použitím Medián filtru

Metody	TP	FP	FN	Přesnost (%)	Citlivost (%)	F1-Skóre (%)	Čas detekce (s)
LBP	86	26	33	76,786	72,269	74,459	8,496
Haar	92	22	27	80,702	77,311	78,97	11,822
HOG	87	253	32	25,588	73,109	37,908	26,465
OpenCV Haar	57	24	62	70,37	47,899	57	17,029
OpenCV HOG	97	109	22	47,087	81,513	59,692	22,279
OpenCV HOG+SVM	92	3	27	96,842	77,311	85,981	21,529

Tabulka 3: Tabulka metod a jejich charakteristik s použitím Gaussián filtru

C Ukázka detekovaných objektů

(a) LBP

(b) Haar

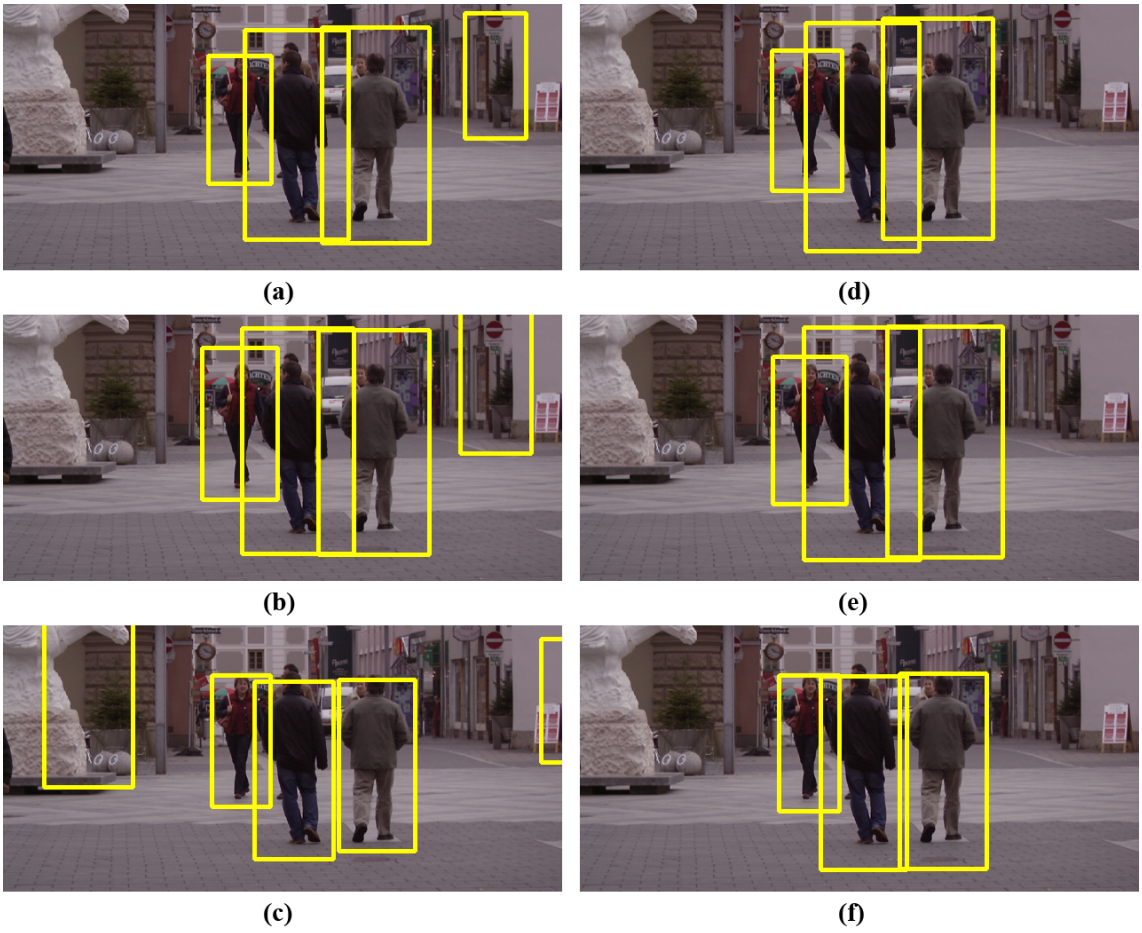
(c) HOG

(d) OpenCV_Haar

(e) OpenCV_HOG

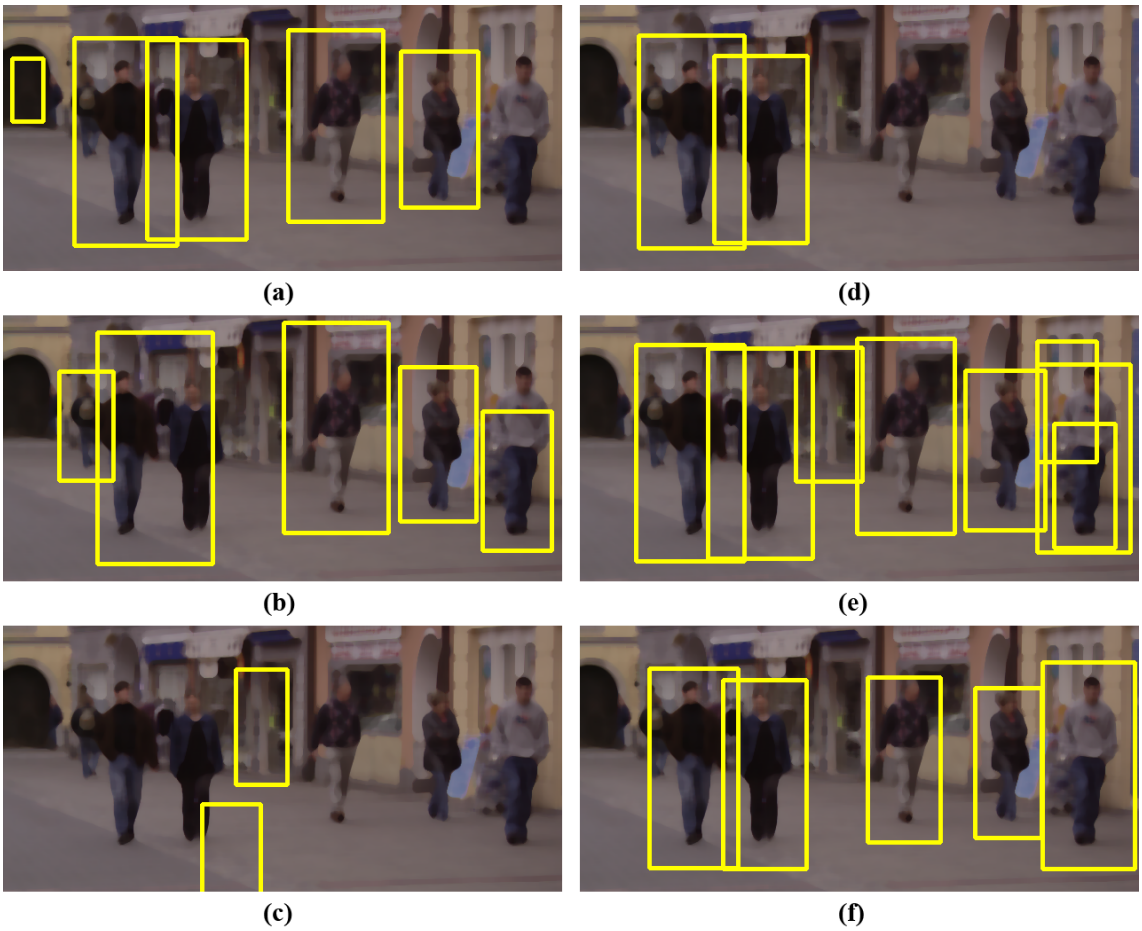
(f) OpenCV_HOG+SVM

C.1 Porovnání příznakových metod



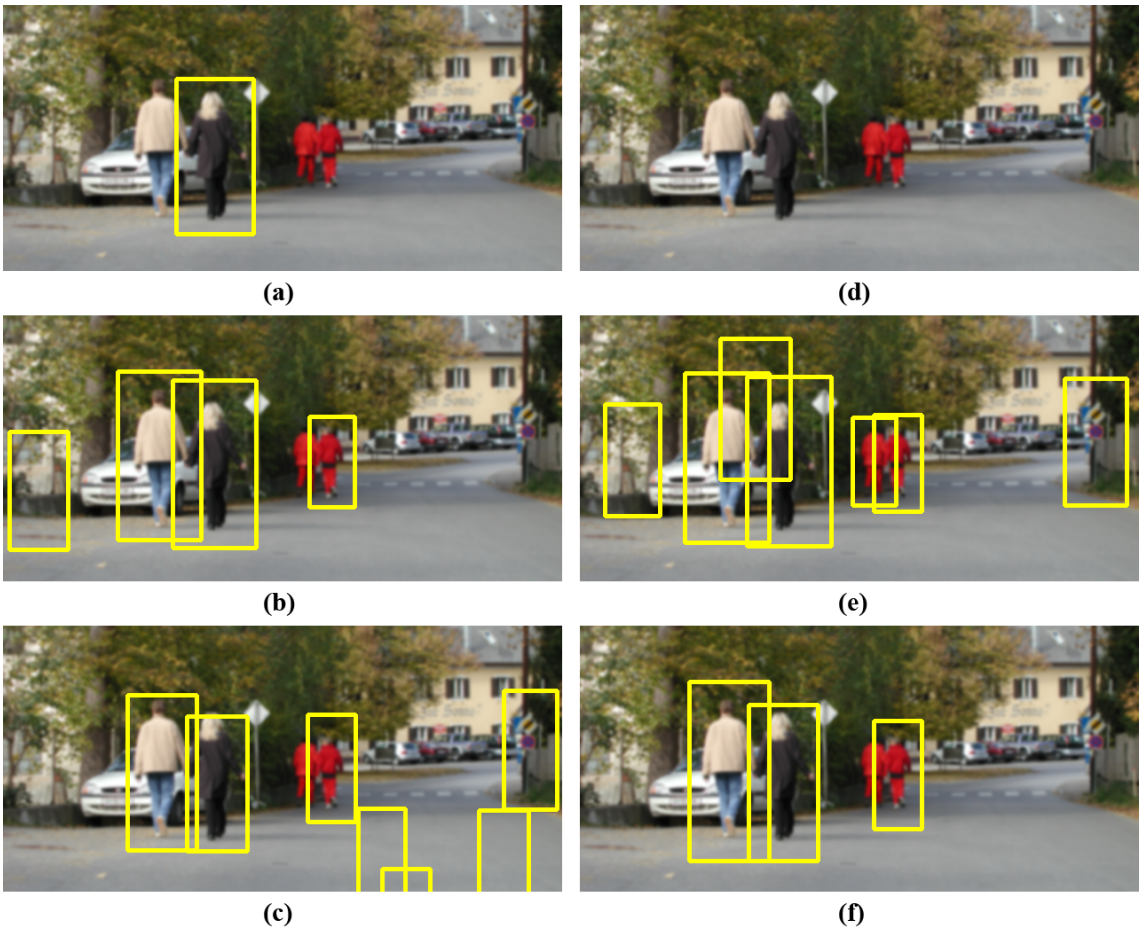
Obrázek 29: Ukázka porovnání metod.

C.2 Porovnání příznakových metod s použitím Medián filtru



Obrázek 30: Ukázka porovnání metod s použitím Medián filtru.

C.3 Porovnání příznakových metod s použitím Gaussián filtru



Obrázek 31: Ukázka porovnání metod s použitím Gaussián filtru.